ПАРАЛЛЕЛЬНОЕ ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ РЕФАЛ

Ю.А. Климов, А.Ю. Орлов

Введение

Существует достаточно широкий класс задач, для решения которых традиционно используются и показывают себя эффективным инструментом функциональные языки программирования. Это задачи в области обработки символьной информации и искусственного интеллекта, постоянно решаемые на грани текущих возможностей, наработок в области теории и практики построения программно-аппаратных комплексов. К классу таких задач относятся анализ и обработка текстов на естественных и искусственных языках, компьютерная алгебра (символьные вычисления), автоматическое доказательство теорем и т.д.

Принципиальное отличие этих задач от вычислительных, успешно решаемых сегодня на суперкомпьютерах, заключается в том, что в связи с их внутренней сложностью, невозможно статически выделить независимые подзадачи, способные выполняться одновременно.

Тем не менее, в связи с растущей доступностью одновременного выполнения вычислений на аппаратном уровне, естественным шагом стало расширение функциональных языков средствами организации параллельных вычислений. Цель этой деятельности заключается в следующем: дать программисту возможность использовать функциональный язык для решения сложных задач, при этом предоставив ему средства организации доступа к ресурсам высокопроизводительных систем для решения больших задач. Важно отметить актуальность проблемы в связи с тем, что волна параллелизма уже захлестывает "рабочие столы" в виде растущей многоядерности и многопроцессорности персональных компьютеров.

Многие используемые на практике функциональные языки программирования расширяются средствами организации параллельных вычислений. Ниже описывается экспериментальная разработка инструмента распараллеливания программ на функциональном языке Рефал [7], которая проводится авторами.

В качестве средства распараллеливания нижнего уровня используется Т-система [3] -- система автоматизированного распараллеливания программ на языках C/C++, разрабатываемая в ИПС РАН.

Язык Рефал Плюс и OpenTS

От других языков Рефал отличается прежде всего своей основной структурой данных, называемой *рефал-выражением*. Рефал-выражение есть последовательность (может быть, пустая) *рефал-термов*. Рефал-терм есть либо некий символ (например, число, буква, ссылка на объект), либо рефал-выражение в скобках. Таким образом, рефал-выражение представляет из себя последовательность символов с правильно расставленными скобками. Надевание скобок является конструктором выражения. Вторым конструктором является операция конкатенации выражений. Интересно отметить, что этот конструктор ассоциативен, что не обычно среди общеизвестных языков программирования.

Язык Рефал Плюс [4] является современным, активно развивающимся диалектом языка Рефал. Инструменты для программирования на нем включают полноценную интегрированную среду разработки на базе платформы Eclipse [5]. Современная реализация Рефала Плюс [2, 6] базируется на компиляции программного кода на Рефале в код на широко доступных языках программирования, таких как C++ и Java. Она имеет модульную, расширяемую архитектуру, что позволило легко добавить выход в язык T++ -- расширение C++, используемое в качестве основного языка программирования в среде OpenTS.

OpenTS -- это открытая версия Т-системы, имеющая многоуровневую архитектуру [3]. OpenTS предназначена для автоматического динамического распараллеливания программ на языке C++, при условии соблюдения программистом некоторых ограничений. А именно, программист должен выделить в программе некоторое количество чисто функциональных блоков, оформив их в виде функций, не имеющих побочных эффектов. Внутри таких блоков разрешается использовать все возможности языка C++ (плюс некоторые дополнительные конструкции языка T++ для более тонкого управления параллелизмом).

Рефал Плюс является функциональным языком программирования, поэтому ограничения системы OpenTS на код автоматически выполняются. В следствие чего возможно применение системы OpenTS в качестве среды исполнения для скомпилированных в T++ рефал-программ.

Средства распараллеливания

Функциональный стиль программирования предполагает оперирование *чистыми* функциями, которые обмениваются данными со средой исполнения только через свои аргументы и результаты, т.е. не используют никакого глобального состояния среды.

На первый взгляд, функциональные программы должны легко распараллеливаться, так как чистые функции можно вычислять независимо. Если бы не один существенный момент: функциональное программирование, в отличие от императивного, предполагает, что вызов функции дешев. Однако, в системах с распределенной памятью вызов функции на другом узле может быть очень дорогим, т.к. необходимо доставить на узел, где будет происходить вычисление, все необходимые для работы данные, а затем получить обратно результат. Отсюда возникает необходимость разделить функции на те, в которых

происходят значительные вычисления, и которые имеет смысл "дорого" вызывать в параллель, и на те, вызов которых должен оставаться дешевым.

В текущей параллельной версии языка Рефал Плюс это разделение производит программист в момент объявления функции. Для этого в язык добавлено ключевое слово "$tfunc", которым обозначается функция (называемая *т-функцией*), производящая существенный объем вычислений, и которую имеет смысл вызывать в параллель. Таким образом, т-функции во время исполнения формируют гранулы параллелизма.

Любая функция в программе на Рефале Плюс принимает на вход несколько рефал-выражений и возвращает несколько рефал-выражений. Если т-функция вызывается в параллель, то ее результаты в месте вызова являются неготовыми выражениями. С неготовыми выражениями можно проделывать некоторые операции, не приводящие к ожиданию их вычисления. Это передача их в качестве аргументов функций, надевание скобок, присваивание. Если же для операции требуется знать содержимое выражения или его длину (конкатенация, сравнение, взятие подвыражения), то вычисление будет приостановлено до тех пор, пока не будет вычислен и передан верхний уровень выражения.

При передаче рефал-выражения между вычислительными процессами, всегда передается целиком верхний уровень выражения (без содержимого скобок). Если затем в вычислении понадобится содержимое скобок, то оно будет передано как отдельное выражение, то есть опять целиком его верхний уровень, и так далее. Такой способ передачи данных дает программисту некоторый контроль на уровне языка над размером передаваемых единовременно данных.

Синхронизация вычислений относительно готовности выражений происходит автоматически на уровне OpenTS.

Пример

При функциональном стиле программирования часто используется шаблон применения некоторой функции ко всем элементам списка (в Рефале -- ко всем термам рефал-выражения). В Рефале Плюс это стандартная функция Map, принимающая на вход ссылку на функцию и список. Покажем, как с помощью конструкции "$tfunc" можно реализовать функцию ParMap, которая разбивает список на несколько подсписков заданной длины и на каждом из них (параллельно) запускает обычную функцию Map.

```
// Standard library
$func Map s.func e.list = e.list;
$func? Left s.n e.list = e.head;
$func? Middle s.n s.m e.list = e.middle;

// ParMap example
$tfunc ParMap s.granule s.func e.list = e.list;

ParMap s.granule s.func e.list, {
   <Left s.granule e.list> :: e.head, <Middle s.granule 0 e.list> :: e.tail,
      <ParMap s.granule s.func e.tail> :: e.tail, <Map s.func e.head> :: e.head,
      e.head e.tail;
   <Map s.func e.list>;
}
```

Функция ParMap принимает на вход длину подсписка s.granule, ссылку на функцию s.func и список e.list. Ее определение состоит из двух предложений, разделенных ";". Знак "::" означает присваивание левой части в правую. Если функции Left удалось отщепить начало списка длины s.granule, то на остатке списка в параллель запускается ParMap, а на отщепленном куске -- (обычным, последовательным образом) Map. Затем полученные результаты конкатенируются. Если же отщепить начало нужной длины не удалось, срабатывает второе предложение, просто запускающее функцию Map.

Важно отметить, что OpenTS производит распределение вычислений автоматически во время работы программы, поэтому значение параметра s.granule может находиться в достаточно широких пределах. Оно должно быть не слишком маленьким, чтобы избежать многочисленных пересылок данных, и не слишком большим, чтобы система смогла равномерно распределить нагрузку.

Направления развития

Функции в программе на Рефале получают на вход рефал-выражения и выдают новые рефал-выражения. Отметим существенный момент. Рефал-выражение -- достаточно простая структура данных. В отличие от C, в Рефале нет указателей. Рефал-выражение, будучи однажды создано, остается неизменным на протяжении всей работы программы.

Это дает возможность собрать осмысленную статистику времени работы функций в зависимости от размера входных данных (например, длины и глубины получаемых рефал-выражений). И, исходя из этого, можно динамически решать, имеет ли смысл запускать данный вызов в параллель.

Другим направлением развития является создание специализированной версии Т-системы, использующей особенности языка. В языке Рефал (как и в других функциональных языках) данные

неизменяемые, поэтому нет необходимости в копировании данных-аргументов с целью их запоминания при отложенном или удаленном вызове функции. Кроме того, если работа происходит в системе с общей памятью, то копирование данных при вызовах функций вообще не нужно. Использование таких оптимизаций должно существенным образом повлиять на эффективность исполнения параллельных программ.

Заключение

Параллельная версия языка Рефал Плюс, так же как и другие средства программирования на Рефале Плюс, общедоступна через веб-сайт проекта [1].

Авторы благодарны С.М. Абрамову, С.А. Романенко и коллективу разработчиков Т-системы за поддержку и консультации при разработке проекта.

Работа выполняется при поддержке РФФИ (проекты № 06-01-00574-а, № 07-07-92100-ГФЕН_а и № 08-07-00280-а) и Роснауки (проект № 2007-4-1.4-18-02-064).

ЛИТЕРАТУРА:

1. Сайт проекта Refal+ http://rfp.botik.ru/.
2. С.М. Абрамов, А.Ю. Орлов, Л.В. Парменова, С.М. Пономарева, А.Ф. Слепухин "Новый подход к реализации системы программирования Рефал Плюс" // Труды международной конференции "Программные системы: теория и приложения", Переславль-Залесский, май 2004, М.: Физматлит, т.1., с. 373-401.
3. С.М. Абрамов, А.И. Адамович, А.В. Инюхин, А.А. Московский, В.А. Роганов, Ю.В. Шевчук, Е.В. Шевчук "Т-система с открытой архитектурой" // Труды международной научной конференции "Суперкомпьютерные системы и их применение", Минск, 26-28 октября 2004 г., ОИПИ НАН Беларуси, с.18-22.
4. Р.Ф. Гурин, С.А. Романенко "Язык программирования Рефал Плюс" // Переславль-Залесский: Изд-во "Университет города Переславля", 2006.
5. Ю.А. Климов, А.Ю. Орлов, С.А. Романенко "Рефал Плюс в среде Eclipse" // Труды научно-практической конференции "Программные системы: теория и приложения", Переславль-Залесский, апрель 2008, Изд-во "Университет города Переславля", 2008, т.1, с.123-132.
6. Ю.А. Климов, А.Ю. Орлов, С.А. Романенко "Язык Рефал Плюс на платформе Java" // Труды Всероссийской научной конференции "Научный сервис в сети Интернет: многоядерный компьютерный мир. 15 лет РФФИ", Новороссийск, 24-29 сентября 2007 г., М.: Изд-во МГУ, 2007, с.207–209.
7. В.Ф. Турчин "Метаязык для формального описания алгоритмических языков" // В сб.: Цифровая вычислительная техника и программирование, М.: Сов. Радио, 1966, с.116-124.